

FILEID**RMOSHARE

F 5

RRRRRRRR RRRRRRRR MM MM 000000 SSSSSSSS HH HH AAAAAA RRRRRRRR EEEEEEEE
RR RR MMMM MMMM 00 00 SS SSSSSSSS HH HH AA AA RR RR RR EE
RR RR MMMM MMMM 00 00 SS SSSSSSSS HH HH AA AA RR RR RR EE
RR RR MM MM MM 00 0000 SS SSSSSSSS HH HH AA AA RR RR RR EE
RR RR MM MM MM 00 0000 SS SSSSSSSS HH HH AA AA RR RR RR EE
RRRRRRRR MM MM 00 00 00 SSSSSS HHHHHHHHHH AA AA RRRRRRRR EEEEEEEE
RRRRRRRR MM MM 00 00 00 SSSSSS HHHHHHHHHH AA AA RRRRRRRR EEEEEEEE
RR RR MM MM 0000 00 SS HH HH AAAAAAAA RR RR EE
RR RR MM MM 0000 00 SS HH HH AAAAAAAA RR RR EE
RR RR MM MM 00 00 SS HH HH AA AA RR RR RR EE
RR RR MM MM 000000 SSSSSSSS HH HH AA AA RR RR RR EE
RR RR MM MM 000000 SSSSSSSS HH HH AA AA RR RR RR EE
....

LL IIIII SSSSSSSS
LL IIIIII SSSSSSSS
LL II SS
LLLLLLLLLL IIIIII SSSSSSSS
LLLLLLLLLL IIIIII SSSSSSSS

RM
VO

(2)	186	DECLARATIONS
(3)	219	RMSINIT_SFSB - Allocate and initialize the SFSB
(4)	388	RMSINIT_SFSB_IRB - Allocate and initialize the SFSB using IRAB.
(5)	483	CHECK_SHARE_OPTIONS
(6)	543	File Locking routines
(7)	710	RMSRLS_SFSB - Deallocate the SFSB
(8)	790	RMSINIT_GBSB - Allocate and init the GBSB
(9)	910	Global Buffer Section locking routines
(10)	1010	RMSRLS_GBSB - Deallocate the GBSB and dequeue the lock on it
(11)	1069	VALIDATE_EBK_HBK

```
0000 1 $BEGIN RMOSHARE,000,RM$RMS0,<SHARING ROUTINES>
0000 2
0000 3 :
0000 4 :*****
0000 5 :*
0000 6 :* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7 :* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8 :* ALL RIGHTS RESERVED.
0000 9 :*
0000 10 :* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11 :* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12 :* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13 :* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14 :* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15 :* TRANSFERRED.
0000 16 :*
0000 17 :* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18 :* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19 :* CORPORATION.
0000 20 :*
0000 21 :* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22 :* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23 :*
0000 24 :*
0000 25 :*****
0000 26 :
```

0000 28 :++
0000 29 .
0000 30 . Facility: RMS32
0000 31 .
0000 32 . Abstract:
0000 33 .
0000 34 . this module is comprised of subroutines which are used by the
0000 35 . file sharing logic of rms. these utilities were written to
0000 36 . facilitate the processing of functions which are called many
0000 37 . times or from many locations. the functions include the
0000 38 . acquisition and release of resources and buffers.
0000 39 .
0000 40 . Environment:
0000 41 .
0000 42 . star processor running starlet exec.
0000 43 .
0000 44 . Author: Keith B. Thompson creation date: 9-Jul-1982
0000 45 .
0000 46 . Modified By:
0000 47 .
0000 48 . V03-036 JEJ0053 J E Johnson 30-Aug-1984
0000 49 . Put in a test to bugcheck if we ever try to store an
0000 50 . EOF of 0/0.
0000 51 .
0000 52 . V03-035 DGB0024 Donald G. Blair 07-Mar-1984
0000 53 . Allocate a fib descriptor and fib in VALIDATE_EBK_HBK
0000 54 . so that we can fill in the FIB\$B_AGENT_MODE field and
0000 55 . pass it to the file system.
0000 56 .
0000 57 . V03-034 JWT0160 Jim Teague 29-Feb-1984
0000 58 . Remove calls to RM\$DEALLEFN.
0000 59 .
0000 60 . V03-033 SHZ0014 Stephen H. Zalewski 23-Sep-1983
0000 61 . Replace line in lock manager call deleted by lja0098.
0000 62 .
0000 63 . V03-032 LJA0098 Laurie J. Anderson 20-Sep-1983
0000 64 . Make sure that when a EFN is allocated that it is also
0000 65 . deallocated even if it IS used. This will fix a problem
0000 66 . RMS will hang a user process in a \$CLOSE if Deferred
0000 67 . write is set. RMS will flush the BDB's using one EFN
0000 68 . and then write the file header characteristics using
0000 69 . another EFN. The user ends up waiting for a EFN which
0000 70 . is never used again.
0000 71 .
0000 72 . This is probably a temporary fix. The real fix will
0000 73 . be done for FT2 of V3B.
0000 74 .
0000 75 . V03-031 SHZ0013 Stephen H. Zalewski 19-Sep-1983
0000 76 . Add a new routine that will initialize the SFSB using an
0000 77 . IRAB instead of an IFAB.
0000 78 .
0000 79 . V03-030 SHZ0012 Stephen H. Zalewski 12-Sep-1983
0000 80 . If a user attempts to open a file shared and specified UFO
0000 81 . in the FOP filed of the FAB, then he must also specify the
0000 82 . UPI bit in the SHR field of the FAB.
0000 83 .
0000 84 . When taking out the shared file lock, get the device name ID

SHARING ROUTINES

J 5

16-SEP-1984 00:37:46 VAX/VMS Macro V04-00
5-SEP-1984 16:22:33 [RMS.SRC]RMOSHARE.MAR;1Page 3
(2)

- 0000 85 : from a new field that will uniquely identify this disk throughout a cluster. (This field is not ascii and is therefore unreadable.)
- 0000 86 :
0000 87 :
0000 88 :
0000 89 :
0000 90 :
0000 91 :
0000 92 :
0000 93 :
0000 94 :
0000 95 :
0000 96 :
0000 97 :
0000 98 :
0000 99 :
0000 100 :
0000 101 :
0000 102 :
0000 103 :
0000 104 :
0000 105 :
0000 106 :
0000 107 :
0000 108 :
0000 109 :
0000 110 :
0000 111 :
0000 112 :
0000 113 :
0000 114 :
0000 115 :
0000 116 :
0000 117 :
0000 118 :
0000 119 :
0000 120 :
0000 121 :
0000 122 :
0000 123 :
0000 124 :
0000 125 :
0000 126 :
0000 127 :
0000 128 :
0000 129 :
0000 130 :
0000 131 :
0000 132 :
0000 133 :
0000 134 :
0000 135 :
0000 136 :
0000 137 :
0000 138 :
0000 139 :
0000 140 :
0000 141 :
V03-029 SHZ0011 Stephen H. Zalewski 10-Aug-1983
Zero the pointer to the global buffer section in the IFB before dequeuing the lock. This is to prevent last chance from attempting to unmap from the section if the process gets stopped while we are dequeuing the lock.
Turn off checking of the sharing fields in the fab.
V03-028 SHZ0010 Stephen H. Zalewski 28-Jul-1983
Add support for cluster global buffers.
V03-027 SHZ0009 Stephen H. Zalewski 26-Jun-1983
Make SFSB a root lock instead of a child of the XQP lock.
V03-026 SHZ0008 Stephen H. Zalewski 25-Jun-1983
Check a different set of journal flags in RM\$RLS_SFSB.
V03-025 SHZ0007 Stephen H. Zalewski 30-Apr-1983
Modify the way register storage is done in VALIDATE_EBK_HBK routine.
V03-024 SHZ0006 Stephen H. Zalewski 18-Apr-1983
Add cluster failover support for file locking.
V03-023 SHZ0005 Stephen H. Zalewski 13-Apr-1983
Do not set IFBV_STALL_LOCK flag around calls to RM\$STALL_LOCK. This flag is now set and cleared in RM\$STALL_LOCK itself. This fixes a deadlock condition with global buffers.
V03-022 SHZ0004 Stephen H. Zalewski 12-Apr-1983
Make sure last accessor to a global buffer section zeroes out all fields in the value block for the GBSB.
V03-021 JWH0199 Jeffrey W. Horn 22-Mar-1983
Save the file lock if we are trying to give it up within a recovery unit.
V03-020 KBT0498 Keith B.Thompson 21-Feb-1983
Fix the file lock for xqp
V03-019 KBT0496 Keith B. Thompson 18-Feb-1983
Fix init_gbsb to stall with the correct structure and put in a temporary hack to fix failover
V03-018 KBT0492 Keith B. Thompson 9-Feb-1983
Check for compatible sharing-fac options
V03-017 KBT0483 Keith B. Thompson 1-Feb-1983
Fix kbt0450 (r9 NOT r10!)
V03-016 KBT0465 Keith B. Thompson 10-Jan-1983
Use parent lock id for file locks if using the xqp and request all of the lock modes to be EXEC

0000	142	
0000	143	V03-015 KBT0450 Keith B. Thompson 6-Jan-1983
0000	144	Load the ifab eof stuff correctly in init_sfsb
0000	145	
0000	146	V03-014 KBT0431 Keith B. Thompson 3-Dec-1982
0000	147	Change the way the shared lock name is made
0000	148	
0000	149	V03-013 KBT0402 Keith B. Thompson 30-Nov-1982
0000	150	Change fwa\$t_shrfildev to fwa\$t_shrfilbuf
0000	151	
0000	152	V03-012 SHZ0003 Stephen H. Zalewski, 29-Oct-1982 19:02
0000	153	Zero the GBH_PTR field in the IFAB after dequeuing the lock
0000	154	on the GBSB. This prevents last chance from attempting to
0000	155	dequeue the GBSB again.
0000	156	
0000	157	V03-011 JWH0105 Jeffrey W. Horn 21-Sep-1982
0000	158	Fix bug in storage of HBK,EBK pair into value block.
0000	159	
0000	160	V03-010 SHZ0002 Stephen H. Zalewski, 21-Sep-1982 13:07
0000	161	Make RM\$RAISE_GBS_LOCK and RMSLOWER_GBS_LOCK stuff the
0000	162	appropriate lock mode into R0. Remove RM\$GET_GBS_LOCK
0000	163	entry point. Make RM\$RLS_SFSB always look at R9 for
0000	164	the IFAB. Removed unnecessary block definitions.
0000	165	
0000	166	V03-009 KBT0324 Keith B. Thompson 10-Sep-1982
0000	167	Removed all S0 sharing code add rm\$rls_sfsb, rename
0000	168	rm\$get_gsb_lock to rm\$init_gbsb, rm\$rls_gbs_lock to
0000	169	rm\$rls_gbsb, add routine rm\$rls_sfsb and make this
0000	170	a new source module
0000	171	
0000	172	V03-008 SHZ0001 Stephen H. Zalewski, 1-Sep-1982 14:44
0000	173	Add global buffer section locking routines.
0000	174	
0000	175	V03-007 KBT0300 Keith B. Thompson 24-Aug-1982
0000	176	Reorganize psects
0000	177	
0000	178	V03-006 KBT0123 Keith B. Thompson 7-Aug-1982
0000	179	Add more locking features
0000	180	
0000	181	V03-005 KBT0079 Keith B. Thompson 9-Jul-1982
0000	182	Add new file locking routines
0000	183	
0000	184	--
0000	185	
0000	186	.SBTTL DECLARATIONS
0000	187	
0000	188	\$ATRDEF : acp attribute list def.
0000	189	\$ENQDEF : enq service definitions
0000	190	\$FABDEF : fab data definition
0000	191	\$FATDEF : ACP QIO record attribute defs.
0000	192	\$FIBDEF : file information block definitions
0000	193	\$FTLDEF : fatal bugcheck codes
0000	194	\$FWADEF : file work area definitions
0000	195	\$GBSBDEF : global buffers synchronization block
0000	196	\$IFBDEF : ifab data definitions
0000	197	\$IODEF : irab data definitions
0000	198	

SHARING ROUTINES
DECLARATIONS

0000	199
0000	200
0000	201
0000	202
0000	203
0000	204
0000	205
0000	206
0000	207
0000	208
0000	209
0000	210
0000	211
0000	212
0000	213
0000	214
0000	215
0000000F	216
00000016	217

L 5

16-SEP-1984 00:37:46 VAX/VMS Macro V04-00
5-SEP-1984 16:22:33 [RMS.SRC]RMOSHARE.MAR;1Page 5
(2)

\$LCKDEF ; enq lock definitions
\$PSLDEF ; access mode definitions
\$RMSDEF ; rms error codes
\$SF\$BDEF ; shared file synchronization block
\$SSDEF

These are the FAC/SHR fields which must be compatible for
inter-process file sharing

NOTES:

- 1) We allow one process to mutistream while another does not
- 2) The NIL option is taken care of in rm\$access

SHRBITS = <FAB\$M_GET!FAB\$M_PUT!FAB\$M_DEL!FAB\$M_UPD>
FHCLEN = IFB\$C_FHAEND-IFB\$B_RFMOORG ; FHC Length

0000 219 .SBTTL RMSINIT_SFSB - Allocate and initialize the SFSB
0000 220
0000 221 :++
0000 222 :
0000 223 RMSINIT_SFSB
0000 224
0000 225 This routine allocates the SFSB. The SFSB is allocated one per IFAB,
0000 226 and is used to contain the necessary local lock manager context for
0000 227 locking the shared file. This will be allocated even when sharing is
0000 228 only multi-stream, as the lock manager will be used in that case also.
0000 229
0000 230 A protected write lock is requested on the file also.
0000 231
0000 232 Calling Sequence:
0000 233
0000 234 BSBW RMSINIT_SFSB
0000 235
0000 236 Input Parameters:
0000 237
0000 238 r10 fwa address
0000 239 r9 ifab address
0000 240 r8 fab address
0000 241
0000 242 Implicit Inputs:
0000 243
0000 244 fwa\$st_fibbuf = used to pick up file id
0000 245 fwa\$sq_device = used to pick up ascii device string
0000 246
0000 247 Output Parameters:
0000 248 r0 status code
0000 249
0000 250 Implicit Outputs:
0000 251
0000 252 sfsb allocated and initialized, a PW lock made on it
0000 253 if an enq error occurs, the \$FAB stv field has the system service
0000 254 code...
0000 255
0000 256 Completion Codes:
0000 257
0000 258 suc, dme, enq, upi and shr
0000 259
0000 260 Side Effects:
0000 261
0000 262 r1-r7 destroyed
0000 263
0000 264 :--
0000 265
0000 266 RMSINIT_SFSB::
0000 267
0000 268
0000 269 : Check to see if we should be doing sharing in the first place
0000 270
0000 271 :
0000 272 BBC #FAB\$V_UFO,FAB\$L_FOP(R8),10\$; ufo can't be spec'd
0000 273 BBS #FAB\$V_UPI,FAB\$B_SHR(R8),10\$; unless upi is also.
0000 274 RMSERR SHR
0000 275

			05	000F	276	RSB			
			60	8F	93	0010	277	10\$: BITB #<FABSM_BIO!FABSM_BRO>,- ; is this any form of block io	
			16	A8	0013	0010	278	FABSB_FAC(R8) ; if so it must be upi	
			0B	OB	13	0015	279	BEQL INIT_SFSB ; no, so don't worry	
06	17	A8	06	E0	0017	0015	280	BBS #FAB\$V_UP, FAB\$B_SHR(R8), INIT_SFSB ; upi? then ok	
					05	001C	281	RMSERR UPI ; get out now	
					05	0021	282	RSB	
						0022	283		
						0022	284		
						0022	285	INIT_SFSB:	
			52	11	9A	0022	286	MOVZBL #<SFSB\$C_BLN/4>, R2 ; get block length to allocate	
			51	59	D0	0025	287	MOVL R9, R1 ; allocate from same page as ifab	
			FFD5	.	30	0028	288	BSBW RM\$GETBLK ; allocate it	
			06	50	E8	002B	289	BLBS R0, 10\$; branch if ok	
					002E	290	RMSERR DMÉ ; dynamic memory exhausted		
78	A9	51	05	0033	291	RSB	292	MOVL R1, IFB\$L_SFSB_PTR(R9) ; return error to caller	
08	A1	10	D0	0034	292	MOVB #SFSB\$C_BID, SFSB\$B_BID(R1) ; address of SFSB in IFAB			
			0038	293	003C	294		; fill in block id	
			003C	294	003C	295			
			003C	295	003C	296		: Make a descriptor in the SFSB, pointing to RESNAM field.	
			003C	296	003C	297		:	
			003C	297	003C	298			
OC	A1	DE	003C	299	MOVAL SFSB\$T_RESNAM(R1), - ; address of RESNAM string				
04	A1		003F	300	SFSB\$L_ADDRESS(R1)	;		to descriptor address field	
			0041	301					
			0041	302					
			0041	303				: Put RMS facility code (RMSS) into first longword of name.	
			0041	304				: Copy File ID from FIB in FWA to next three words of the SFSB.	
			0041	305				: Copy ascii device name from FWA into SFAB after File ID.	
			0041	306				:	
			0041	307					
50	01F4	CA	DE	0041	308	MOVAL FWAST_FIBBUF(R10), R0	;	get address of FIB	
	0A		9B	0046	309	MOVZBW #SFSB\$C_FIX_LEN, -	;	initialize length of RESNAM string	
	61			0048	310	SFSB\$W_NAME_LEN(R1)	;	to fixed length portion	
OC	A1	24534D52	8F	D0	0049	312	MOVL #^A/RMSS/, SFSB\$L_FAC_CODE(R1)	;	set RMS facility code into name
	04	A0	B0	0051	313	0051			
	10	A1	B0	0051	314	MOVW FIB\$W_FID_NUM(R0), -	;		
	06	A0	B0	0054	315	SFSB\$W_FID_NUM(R1)	;	first word of FID	
	12	A1	B0	0056	316	MOVW FIB\$W_FID_SEQ(R0), -	;		
	08	A0	B0	0059	317	SFSB\$W_FID_SEQ(R1)	;	second word of FID	
	14	A1	B0	005B	318	MOVW FIB\$W_FID_RVN(R0), -	;		
			005E	319	0060	SFSB\$W_FID_RVN(R1)	;	last word of FID	
			0060	320					
0198	CA	A0	0060	321	ADDW2 FWASQ_SHRFIL_LCK(R10), -	;	get length of device string		
	61		0064	322	SFSB\$W_NAMELEN(R1)	;	add it to descriptor length		
0198	CA	28	0065	323	MOVC3 FWASQ_SHRFIL_LCK(R10), -	;	move the string to the SFSB		
019C	DA		0069	324	0FWASQ_SHRFIL_LCK+4(R10), -	;	resnam field to name the		
16	A1		006C	325	SFSB\$T_DEV_NAM(R1)	;	file		
			006E	326					
54	78	A9	D0	006E	327	MOVL IFB\$L_SFSB_PTR(R9), R4	;	restore SFSB address into R4	
			0072	328					
			0072	329					
			0072	330				: Lock the shared file protected write. This gives a parent_id to any	
			0072	331				subsequent record locks.	
			0072	332					

00000000'EF 01 16 BA 0072 333 DO_ENQ: JSB RMSSETEFN ; Allocate event flag
 0072 334 POPR #^M<R0> ; Get the flag
 007A 335 SENQ_S LKMODE = #LCK\$K_PWMODE,-
 007A 336 ACMODE = #PSL\$C_EXEC,-
 007A 337 EFN = R0,-
 007A 338 ASTADR = RM\$STALLAST,-
 007A 339 ASTPRM = R9,-
 007A 340 LKSBN = SF\$BSL_LKSBN(R4),-
 007A 341 RESNAM = SF\$BSQ_FILENAME(R4),-
 007A 342 FLAGS = #LCK\$M_VALBLK!LCK\$M_SYNCSTS!LCK\$M_SYSTEM
 009C 343
 009C 344
 0689 29 50 E9 009C 345 BLBC R0,40\$; Error?
 8F 50 B1 009F 346 CMPW R0,#SSS_SYNCH ; Did it complete?
 03 13 00A4 347 BEQL 20\$; Yes, so do not stall
 FF57' 30 00A6 348 BSBW RM\$STALL_LOCK ; Stall for the lock
 50 2C A4 3C 00A9 349 20\$: MOVZWL SF\$BSW_STATUS(R4),R0 ; Get final status
 18 50 E9 00AD 350 BLBC R0,40\$; Did it work?
 00B0 351
 00B0 352 :
 00B0 353 : Check to see if we are first accessor (value block is zero). If so, do not
 00B0 354 : disturb ebk/hbk marks in ifab. If not zero, move values from value block to
 00B0 355 : ebk/hbk fields in ifab.
 00B0 356 :
 00B0 357 :
 00B0 358 ASSUME IFBSL_EBK EQ <IFBSL_HBK + 4>
 00B0 359 ASSUME SF\$BSL_EBK EQ <SF\$BSL_HBK + 4>
 00B0 360
 51 40 A4 3C A4 C1 00B0 361 ADDL3 SF\$BSL_HBK(R4),SF\$BSL_EBK(R4),R1; check for EBK=HBK=0
 05 13 00B6 362 BEQL 30\$; they were zero
 70 A9 3C A4 7D 00B8 363 25\$: MOVQ SF\$BSL_HBK(R4),IFBSL_HBK(R9) ; store eof
 00BD
 0A A4 04 90 00BD 364
 30 A4 D0 00C1 365 30\$: MOVB #LCK\$K_PWMODE,SFSB\$B_CURMODE(R4); Save the current lock value
 0080 C9 00C4 366 MOVL SF\$BSL_LOCK_ID(R4),- ; Save parent id for bucket locks
 00C7 367 IFBSL_PAR_LOCK_ID(R9)
 05 00C7 368 ;BSBB CHECK_SHARE_OPTIONS ; do final checking
 00C8 369 RSB ; return to caller
 00C8 370
 00C8 371
 00C8 372 :
 00C8 373 : An error occurred on the ENQ. Check to see if we can recover from it.
 00C8 374 :
 00C8 375 :
 00C8 376 40\$: ; Was it deadlock?
 50 0E0A 8F B1 00C8 377 CMPW #SSS_DEADLOCK,R0 ; Try it again if it was.
 A3 13 00CD 378 BEQL DO_ENQ ; Did Lock manager gave us old value
 50 09F0 8F B1 00CF 379 CMPW #SSS_VALNOTVALID,R0 ; No, go map the error.
 0B 12 00D4 380 BNEQ 45\$; Yes, validate the data.
 00000001'EF 16 00D6 381 JSB VALIDATE_EBK_HBK ; Continue if successful
 D9 50 E8 00DC 382 BLBS R0,25\$; else map error.
 05 11 00DF 383 BRB 50\$; default to ENQ error for RMSMAPERR
 FF17' 30 00E1 384 45\$: RMSERR ENQ,R1 ; go map the error
 05 00E6 385 50\$: BSBW RMSMAPERR ; and return
 00E9 386 RSB

00EA 388 .SBTTL RM\$INIT_SFSB_IRB - Allocate and initialize the SFSB using IRAB.
00EA 389
00EA 390 :++
00EA 391 : RMS\$INIT_SFSB_IRB
00EA 392 :
00EA 393 :
00EA 394 : This routine allocates the SFSB using an irab to stall on. This only
00EA 395 : occurs in connect when the original sfsb lock has been given away to
00EA 396 : become a system lock for a global buffer section.
00EA 397 : The SFSB is allocated one per IFAB,
00EA 398 : and is used to contain the necessary local lock manager context for
00EA 399 : locking the shared file. This will be allocated even when sharing is
00EA 400 : only multi-stream, as the lock manager will be used in that case also.
00EA 401 :
00EA 402 : A protected write lock is requested on the file also.
00EA 403 :
00EA 404 : Calling Sequence:
00EA 405 :
00EA 406 : BSBW RM\$INIT_SFSB_IRB
00EA 407 :
00EA 408 : Input Parameters:
00EA 409 :
00EA 410 : r10 ifab address
00EA 411 : r9 irab address
00EA 412 :
00EA 413 : Implicit Inputs:
00EA 414 :
00EA 415 : none
00EA 416 :
00EA 417 : Output Parameters:
00EA 418 :
00EA 419 : r0 status code
00EA 420 :
00EA 421 : Implicit Outputs:
00EA 422 :
00EA 423 : sfsb allocated and initialized, a PW lock made on it
00EA 424 : if an enq error occurs, the \$FAB stv field has the system service
00EA 425 : code...
00EA 426 :
00EA 427 : Completion Codes:
00EA 428 :
00EA 429 : suc, dme, enq, upi and shr
00EA 430 :
00EA 431 : Side Effects:
00EA 432 :
00EA 433 : r1-r7 destroyed
00EA 434 :
00EA 435 :--
00EA 436 :
00EA 437 : RMS\$INIT_SFSB_IRB::
00EA 438 :
00EA 439 :
00EA 440 : Lock the shared file protected write. This gives a parent_id to any
00EA 441 : subsequent record locks.
00EA 442 :
00EA 443 :
00EA 444 10\$: JSB RM\$SETEFN ; Allocate event flag

```

01 BA 00F0 445      POPR #^M<R0>          : Get the flag
      00F2 446      SENQ_S LKMODE = #LCK$K_PWMODE,-
      00F2 447      ACMODE = #PSLSC_EXEC,-
      00F2 448      EFN = R0,-
      00F2 449      ASTADR = RM$STALLLAST,-
      00F2 450      ASTPRM = R9,-
      00F2 451      LKSB = SF$BSL_LKSB(R4),-
      00F2 452      RESNAM = SF$BSQ_FILENAME(R4),-
      00F2 453      FLAGS = #LCK$M_VALBLK!LCK$M_SYNCSTS!LCK$M_SYSTEM
      0114 454

0689 8F 1B 50       E9 0114 455      BLBC R0,40$          : Error?
      50     B1 0117 456      CMPW R0#SSS_SYNCH          : Did it complete?
      03     13 011C 457      BEQL 20$               : Yes, so do not stall
      FEDF' 30 011E 458      BSBW RM$STALL_LOCK          : Stall for the lock
      50 2C A4 3C 0121 459 20$: MOVZWL SFSBSW_STATUS(R4),R0
      0A 50  E9 0125 460      BLBC R0,40$          : Get final status
      70 AA 3C A4 7D 0128 461 25$: MOVQ SF$BSL_HBK(R4),IFBSL_HBK(R10)
      0A A4 04 90 012D 462      MOVB #LCK$K_PWMODE,SFSBSB_CURMODE(R4)
      05 0131 463      RSB   : Store EOF.
      0132 464
      0132 465
      0132 466 : An error occurred on the ENQ. Check to see if we can recover from it.
      0132 467
      0132 468 :
      0132 469
      0132 470 40$:
      50 0E0A 8F B1 0132 471      CMPW #SSS_DEADLOCK,R0
      B1 13 0137 472      BEQL 10$               : Was it deadlock?
      50 09F0 8F B1 0139 473      CMPW #SSS_VALNOTVALID,R0
      0B 12 013E 474      BNEQ 45$               : Try it again if it was.
      00000001'EF 16 0140 475      JSB  VALIDATE_EBK_HBK
      DF 50  E8 0146 476      BLBS R0,25$          : Did Lock manager gave us old value
      05 11 0149 477      BRB  50$               : No, go map the error.
      FEAD' 30 0150 478 45$: RMSERR ENQ,R1
      05 0153 480 50$: BSBW RMSMAPERR          : Yes, validate the data.
      0154 481      RSB   : Continue if successful
                           : else map error.
                           : default to ENQ error for RMSMAPERR
                           : go map the error
                           : and return

```

SHARING ROUTINES
CHECK_SHARE_OPTIONS

```

0154 483 .SBTTL CHECK_SHARE_OPTIONS
0154 484
0154 485 :++
0154 486
0154 487 :CHECK_SHARE_OPTIONS
0154 488
0154 489 This routine checks the sharing options in order to see if
0154 490 they are valid. If we are the first in then simply stuff the
0154 491 fields and return.
0154 492
0154 493 Input Parameters:
0154 494
0154 495 r4 - sfsb
0154 496 r8 - fab
0154 497 r9 - ifab
0154 498
0154 499 Output Parameters:
0154 500 none
0154 501
0154 502 Routine Value:
0154 503
0154 504 suc or shr
0154 505
0154 506 Side Effects:
0154 507
0154 508 r1 destroyed
0154 509
0154 510 ;--
0154 511
0154 512 CHECK_SHARE_OPTIONS:
0154 513
0154 514 ASSUME SFSB$B_SHR EQ SFSB$B_FAC+1
0154 515
34 A4 B5 0154 516 TSTW SFSB$B_FAC(R4) ; are we the first in?
07 12 0157 517 BNEQ 10$ ; No, so do checks
0159 518
0159 519
0159 520 ASSUME FAB$B_SHR EQ FAB$B_FAC+1
0159 521 ASSUME SFSB$B_SHR EQ SFSB$B_FAC+1
34 A4 16 A8 B0 0159 522 MOVW FAB$B_FAC(R8),SFSB$B_FAC(R4) ; save the flags for the next guy
16 11 015E 523 BRB SHREXT ; get out
0160 524
0160 525 ASSUME FAB$V_PUT EQ FAB$V_SHRPUT
0160 526 ASSUME FAB$V_GET EQ FAB$V_SHRGET
0160 527 ASSUME FAB$V_DEL EQ FAB$V_SHRDEL
0160 528 ASSUME FAB$V_UPD EQ FAB$V_SHRUPD
0160 529
50 16 A8 35 A4 88 0160 530 10$: BICB3 SFSB$B_SHR(R4),FAB$B_FAC(R8),R0 ; is our access compatible?
50 OF 93 0166 531 BITB #SHRBITS,R0 ; are they different
50 OF 12 0169 532 BNEQ SHRERR ; yes, then error
50 34 A4 17 A8 8B 016B 533 BICB3 FAB$B_SHR(R8),SFSB$B_FAC(R4),R0 ; is their access compatible?
50 OF 93 0171 534 BITB #SHRBITS,R0 ; are any different
04 12 0174 535 BNEQ SHRERR ; yes, then exit
0176 536 SHREXT: RMSSUC ; exit success!
05 0179 537 RSB
017A 538
017A 539 SHRERR: RMSERR INCOMPSPHR ; this is an error

```

RMOSHARE
V04-000

SHARING ROUTINES
CHECK_SHARE_OPTIONS

05 017F 540
0180 541

F 6

16-SEP-1984 00:37:46 VAX/VMS Macro V04-00
5-SEP-1984 16:22:33 [RMS.SRC]RMOSHARE.MAR;1

Page 12
(5)

RSB

; return the bad news

RM
VO

0180 543 .SBTTL File locking routines
 0180 544
 0180 545 :++
 0180 546
 0180 547 RMSRESTORE_LOCK
 0180 548 RMSRAISE_LOCK
 0180 549 RMSLOWER_SYSLOCK
 0180 550 RMSLOWER_LOCK
 0180 551
 0180 552 These routines modify the file lock mode.
 0180 553
 0180 554 Calling Sequence:
 0180 555
 0180 556 BSBW RMSRESTORE_LOCK - Restore the lock mode
 0180 557 BSBW RMSRAISE_LOCK - Get a protected write lock on the file
 0180 558 BSBW RMSLOWER_SYSLOCK - Convert file lock to a system lock (held at NL)
 0180 559 BSBW RMSLOWER_LOCK - Get a concurrent read lock on the file
 0180 560
 0180 561 Input Parameters:
 0180 562
 0180 563 r9 ifab/irab address
 0180 564
 0180 565 Implicit Inputs:
 0180 566 none
 0180 567
 0180 568 Output Parameters:
 0180 569
 0180 570 r0 status code
 0180 571
 0180 572 Implicit Outputs:
 0180 573 none
 0180 574
 0180 575 Completion Codes:
 0180 576
 0180 577 suc, enq
 0180 578
 0180 579 Side Effects:
 0180 580
 0180 581 The ifab eof information is stored in resource block (lower lock)
 0180 582 or updated from the resource block (raise lock)
 0180 583 Could stall
 0180 584
 0180 585 :--
 0180 586
 0180 587 RMSRESTORE_LOCK:: ; Restore the previous lock mode
 043E 8F BB 0180 588 PUSHR #^M<R1,R2,R3,R4,R5,R10>
 50 01 CE 0184 589 MNEG L #1,R0 ; Signal this is a restore
 53 08 DO 0187 590 MOVL #LCK\$M_SYNCSTS!LCK\$M_CONVERT!LCK\$M_VALBLK,R3
 26 11 018A 591 BRB SET_LOCK
 018C 592
 018C 593 RMSRAISE_LOCK:: ; Get a protected write lock on the
 043E 8F BB 018C 594 PUSHR #^M<R1,R2,R3,R4,R5,R10>
 50 04 DO 0190 595 MOVL #LCK\$K_PMODE,R0 ; Stuff the lock mode in R0
 53 08 DO 0193 596 MOVL #LCK\$M_SYNCSTS!LCK\$M_CONVERT!LCK\$M_VALBLK,R3
 1A 11 0196 597 BRB SET_LOCK
 0198 598
 0198 599 RMSLOWER_SYSLOCK:: ; Convert lock to system lock at NL.

043E 8F BB 0198 600 PUSHR #^M<R1,R2,R3,R4,R5,R10>
 50 00 D0 019C 601 MOVL #LCK\$K_NLMODE,R0 ; Stuff lock mode in R0.
 53 0000004B 8F D0 019F 602 MOVL #LCK\$M_SYNCSTS!LCK\$M_CONVERT!LCK\$M_VALBLK!LCK\$M_CVTSYS,R3
 OA 11 01A6 603 BRB SET_LOCK
 01A8 604
 RMSLOWER LOCK::
 043E 8F BB 01A8 605 PUSHR #^M<R1,R2,R3,R4,R5,R10> ; Get a concurrent read lock on the
 50 01 D0 01AC 606 MOVL #LCK\$K_CRMODE,R0 ; Stuff the lock mode in R0
 53 0B D0 01AF 607 MOVL #LCK\$M_SYNCSTS!LCK\$M_CONVERT!LCK\$M_VALBLK,R3
 01B2 608
 01B2 609
 01B2 610 SET_LOCK:
 5A 59 D0 01B2 611 MOVL R9,R10 ; Move ifab into r10 for stall
 01B5 612
 01B5 613 ASSUME <IFBS\$C_BID&1> EQ 1 ; in case it's really irab
 01B5 614 ASSUME <IRBS\$C_BID&1> EQ 0
 01B5 615 ASSUME IFBS\$B_BID EQ IRBS\$B_BID
 01B5 616
 03 08 AA E8 01B5 617 BLBS IFBS\$B_BID(R10),1\$; Do we have a ifab or irab
 5A 6A D0 01B9 618 MOVL IRBS\$L_IFAB_LNK(R10),R10 ; Get ifab
 54 78 AA D0 01BC 619 1\$: MOVL IFBS\$L_SFSB_PTR(R10),R4 ; Get sfsb
 63 13 01C0 620 BEQL ENQSUC ; If the file is not shared
 01C2 621
 01C2 622
 55 50 D0 01C2 623 MOVL R0,R5 ; Save lock value
 04 18 01C5 624 BGEQ 10\$; Branch if not restore
 55 0B A4 90 01C7 625 MOVB SFSB\$B_PREMODE(R4),R5 ; Restore previous mode
 01CB 626
 01CB 627 ; Save the current mode of the lock in case this is a conversion
 01CB 628 ; for a I/O stall
 01CB 629
 01CB 630 ;
 01CB 631
 0B A4 0A A4 90 01CB 632 10\$: MOVB SFSB\$B_CURMODE(R4),SFSB\$B_PREMODE(R4)
 01D0 633
 01D0 634 ; See if we already have the lock being requested, if so exit
 01D0 635 ; NOTE: If someone expects to get current information in the
 01D0 636 ; lock value block by requesting a lock of the same mode
 01D0 637 ; will be disappointed by this.
 01D0 638
 01D0 639
 01D0 640
 0A A4 55 91 01D0 641 CMPB R5,SFSB\$B_CURMODE(R4)
 4F 13 01D4 642 BEQL ENQSUC
 01D6 643
 01D6 644 ; Store the eof info into the value block
 01D6 645
 01D6 646
 01D6 647
 01D6 648 ASSUME IFBS\$L_EBK EQ <IFBS\$L_HBK + 4>
 01D6 649 ASSUME SFSB\$E_EBK EQ <SFSB\$E_HBK + 4>
 01D6 650
 3C A4 70 AA 7D 01D6 651 MOVQ IFBS\$L_HBK(R10),SFSB\$L_HBK(R4)
 50 13 01DB 652 BEQL BAD_EBKHBK
 01DD 653
 01DD 654 ; Do the ENQ conversion.
 01DD 655
 01DD 656 ;

```

        01DD 657
FE20' 30 01DD 658 ENQ: BSBW RM$SETEFN
          01 BA 01E0 659 POPR #^M<R0>
          01E2 660 SENQ_S EFN = R0,-
          01E2 661 LKMODE = R5,-
          01E2 662 LKSB = SF$BSL_LKSB(R4),-
          01E2 663 FLAGS = R3,-
          01E2 664 ASTADR = RM$STALLAST,-
          01E2 665 ASTPRM = R9
          0202 666

        36 50 E9 0202 667 BLBC R0,ENQERR
0689 BF 50 B1 0205 668 CMPW R0,#SSS_SYNCH
          07 13 020A 669 BEQL 10$ : We ok?
          08 BB 020C 670 PUSHR #^M<R3> : Did it complete?
FDEF' 30 020E 671 BSBW RM$STALL_LOCK : Yes, so do not stall
          08 BA 0211 672 POPR #^M<R3> : Save ENQ flags around stall
          50 2C A4 3C 0213 673 10$: MOVZWL SF$BSW STATUS(R4),R0 : Stall for the lock
          21 50 E9 0217 674 BLBC R0,ENQERR : Restore ENQ flags.
          021A 675 : Get the final status
          021A 676 : Branch on failure
          021A 677
          021A 678 : Store the eof info into the ifb from the value block
          021A 679 :
          021A 680
          021A 681 ASSUME IFB$L_EBK EQ <IFB$L_HBK + 4>
          021A 682 ASSUME SF$BS$[_EBK] EQ <SF$BS$[_HBK] + 4>
          021A 683
          70 AA 3C A4 7D 021A 684 HBKEBK: MOVQ SF$BSL_HBK(R4),IFB$L_HBK(R10)
          13 13 021F 685 BEQL BAD_EBKHBK1
          0221 686
          0A A4 55 90 0221 687 ALTSUC: MOVB R5,SFSB$B_CURMODE(R4) : Save the current lock value
          0225 688 ENQSUC: RMSSUC : Signal success
          043E 8F 0228 689 ENQRET: POPR #^M<R1,R2,R3,R4,R5,R10> : Restore registers
          05 022C 690 RSB : Return
          022D 691
          022D 692 BAD_EBKHBK:
          022D 693 RMSPBUG FTLS_BADEBKHBK
          0234 694 BAD_EBKHBK1:
          0234 695 RMSPBUG FTLS_BADEBKHBK
          023B 696
          023B 697 ENQERR:
          50 0E0A 8F B1 023B 698 CMPW #SSS_DEADLOCK,R0 : Was it deadlock?
          9B 13 0240 699 BEQL ENQ : Try it again if it was.
          50 09F0 8F B1 0242 700 CMPW #SSS_VALNOTVALID,R0 : Did Lock manager gave us old value
          0B 12 0247 701 BNEQ 10$ : No, go map error.
          00000001'EF 16 0249 702 JSB VALIDATE_EBK_HBK : Yes, get correct hbk/ebk values.
          CB 50 E8 024F 703 BLBS R0,HBKEBR
          05 11 0252 704 BRB 20$ : default to ENQ error for RMSMAPERR
          FDA4' 30 0254 705 10$: RMSERR ENQ,R1 : go map the error
          FFC9 31 0259 706 20$: BSBW RMSMAPERR : and return
          025C 707 BRW ENQRET
          025F 708

```

```

025F 710 .SBTTL RM$RLS_SFSB - Deallocate the SFSB
025F 711
025F 712 ++
025F 713
025F 714 RM$RLS_SFSB
025F 715
025F 716 This routine deallocates the SFSB and dequeues the file lock.
025F 717
025F 718 Calling Sequence:
025F 719
025F 720 BSBW RM$RLS_SFSB
025F 721
025F 722 Input Parameters:
025F 723
025F 724 r9 ifab address
025F 725
025F 726 Implicit Inputs:
025F 727
025F 728 none
025F 729
025F 730 Output Parameters:
025F 731
025F 732 none
025F 733
025F 734 Implicit Outputs:
025F 735
025F 736 none
025F 737
025F 738 Completion Codes:
025F 739
025F 740 none
025F 741
025F 742 Side Effects:
025F 743
025F 744 none
025F 745
025F 746 --
025F 747
025F 748 RM$RLS_SFSB::: 12 BB
025F 749 PUSHR #^M<R1,R4> ; save work registers
54 78 A9 D0 0261 750
      28 13 0261 751 MOVL IFBSL_SFSB_PTR(R9),R4 ; get address of sfsb
      0265 752 BEQL 30$ ; exit if none
      0267 753
      0267 754
      0267 755 : Store the eof info into the value block
      0267 756 :
      0267 757
      0267 758 ASSUME IFBSL_EBK EQ <IFBSL_HBK + 4>
      0267 759 ASSUME SFSB$C_EBK EQ <SFSB$C_HBK + 4>
      0267 760
      3C A4 70 A9 7D 0267 761 MOVQ IFBSL_HBK(R9),SFSB$L_HBK(R4)
      026C 762
      026C 763
      026C 764 : If a recovery-unit is active on this file then save the file lock
      026C 765
      026C 766

```

08 00A2 C9 02 E1 026C 767 BBC #IFBSV RUP,IFBSB_JNLFLG2(R9),10\$; branch if no RU in prog
FF33 30 0272 768 BSBW RM\$LOWER_LOCK ; lower the lock to correct mode
FD88' 30 0275 769 BSBW RMSSAVE_FL ; go save the file lock
OF 11 0278 770 BRB 20\$
027A 771
027A 772 ;
027A 773 ; Deq the lock
027A 774 ;
027A 775
027A 776 10\$: \$DEQ_S LKID = SFSBSL_LOCK_ID(R4),-
027A 777 VALBLK = SFSBSL_LVB(R4)
0289 778
0289 779 ;
0289 780 ; Deallocate the sfsb
0289 781 ;
0289 782
FD74' 30 0289 783 20\$: BSBW RM\$RELBLK1 ; address in r4
78 A9 D4 028C 784 CLRL IFBSL_SFSB_PTR(R9) ; clear the ifab pointer to it
028F 785
12 BA 028F 786 30\$: POPR #^M<R1,R4> ; restore registers
05 0291 787 RSB ; return to caller
0292 788

0292 790 .SBTTL RMS\$INIT_GBSB - Allocate and init the GBSB
 0292 791
 0292 792 ++
 0292 793 RMS\$INIT_GBSB
 0292 795
 0292 796 This routine allocates the GBSB. The GBSB is allocated one per IFAB,
 0292 797 and is used to contain the necessary local lock manager context for
 0292 798 locking a global section used for global buffers.
 0292 799
 0292 800 An exclusive lock is requested on the global section.
 0292 801
 0292 802 Calling Sequence:
 0292 803
 0292 804 BSBW RMS\$INIT_GBSB
 0292 805
 0292 806 Input Parameters:
 0292 807
 0292 808 r10 ifab address
 0292 809 r9 irab address
 0292 810
 0292 811 Implicit Inputs:
 0292 812
 0292 813 none
 0292 814
 0292 815 Output Parameters:
 0292 816
 0292 817 r0 status code
 0292 818
 0292 819 Implicit Outputs:
 0292 820
 0292 821 gbsb allocated and initialized, an EX lock made on it
 0292 822 if an enq error occurs, the \$FAB stv field has the system service
 0292 823 code...
 0292 824
 0292 825 Completion Codes:
 0292 826
 0292 827 suc, dme, enq
 0292 828
 0292 829 Side Effects:
 0292 830
 0292 831 None...
 0292 832
 0292 833 --
 0292 834
 0292 835 RMS\$INIT_GBSB::
 52 00FE 8F BB 0292 836 PUSHR #^M<R1,R2,R3,R4,R5,R6,R7> ; save work registers
 44 8F 9A 0296 837 MOVZBL #GBSB\$C_BLN,R2 ; get block length to allocate
 51 5A D0 029A 838 MOVL R10,R1 ; allocate from same page as ifab
 FD60. 30 029D 839 BSBW RMS\$GETSPC ; allocate it
 08 50 E8 02A0 840 BLBS R0,1\$; branch if ok
 0078 31 02A8 841 RMSERR DMÉ ; dynamic memory exhausted
 7C AA 51 D0 02AB 842 BRW 25\$; return error to caller
 1109 8F B0 02AF 843 1\$: MOVL R1,IFBSL_GBSB_PTR(R10) ; address of GBSB in IFAB
 08 A1 02B3 844 ASSUME GBSB\$B_B[N EQ GBSB\$B_BID+1
 08 AF 845 MOVW #GBSB\$C_BID+<GBSB\$C_BLN@6>,- GBSB\$B_BID(R1) ; fill in block length and id
 08 A1 02B3 846 GBSB\$B_BID(R1)

			02B5	847							
			02B5	848							
			02B5	849	;	Make a descriptor of the first two longwords in GBSB, pointing to RESNAM					
			02B5	850	;	field. Fill RESNAM with copy of RESNAM from SFSB.					
			02B5	851	:						
			02B5	852							
			OC A1 DE	02B5	853	MOVAL	GBSBST_RESNAM(R1),-	;	Move address of RESNAM to descriptor		
			04 A1	02B8	854		GBSB\$L_ADDRESS(R1)				
52	78 AA DO	02BA	855			MOVL	IFBSL_SFSB_PTR(R10),R2	;	Move SFSB address to R2.		
	62 B0 02BE	856				MOVW	SFSBSW_NAME_LEN(R2),-	;	Move length of RESNAM into desc.		
	61 02C0	857					GBSB\$W_NAME_LEN(R1)				
	61 28 02C1	858				MOVC3	GBSB\$W_NAME_LEN(R1),-	;			
	OC A2 02C3	859					SFSB\$T_RESNAM(R2),-	;	Move the SFSB RESNAM to the GBSB		
	OC A1 02C5	860					GBSBST_RESNAM(R1)	;	resnam field to name the lock.		
	02C7	861									
54	7C AA DO	02C7	862			MOVL	IFBSL_GBSB_PTR(R10),R4	;	restore GBSB address into R4		
	02CB	863									
00000000'GF	D5	02CB	864			TSTL	G^EXE\$GL_SYSID_LOCK	;	Make sure we have a parent lock.		
07	12	02D1	865			BNEQ	5\$;	Yes, continue		
	02D3	866					RMSPBUG FTL\$_NOPARENT	;	No parent, boom....		
	02DA	867									
	02DA	868									
	02DA	869	;	Lock the global section for exclusive access.							
	02DA	870	:								
	02DA	871									
00000000'EF	16	02DA	872	5\$: JSB	RMSSETEFN						
01	BA	02E0	873	POPR	#^M<R0>				;	Allocate event flag	
		02E2	874							;	Get the flag
		02E2	875								
		02E2	876								
		02E2	877								
		02E2	878								
		02E2	879								
		02E2	880								
		02E2	881								
		02E2	882								
		0308	883								
0689 8F 1D 50	E9	0308	884	BLBC	R0,40\$						
	50 B1	030B	885	CMPW	R0,\$SSS_SYNCH						
	03 13	0310	886	BEQL	10\$						
FCEB'	30	0312	887	BSBW	RMSSTALL_LOCK						
		0315	888								
50 2C A4	3C	0315	889	10\$:							
	0C 50	E9	0319	890	MOVZWL	GBSB\$W_STATUS(R4),R0					
0A A4 05	90	031C	891	BLBC	R0,40\$						
		0320	892	MOVBL	#LCK\$K_EXMODE,GBSB\$B_CURMODE(R4)						
00FE 8F	BA	0323	893	RMSSUC							
	05	0327	894	25\$:	POPR	#^M<R1,R2,R3,R4,R5,R6,R7>					
		0328	895	RSB							
		0328	896								
		0328	897		;	An error occurred on the ENQ. See if we can recover from it.					
		0328	898		:						
		0328	899								
50 0EOA 8F	B1	0328	900	40\$:							
	AB 13	032D	901	CMPW	#SSS_DEADLOCK,R0						
50 09F0 8F	B1	032F	902	BEQL	5\$						
		903		CMPW	#SSS_VALNOTVALID,R0						

: Was it deadlock?
 : Try it again if it was.
 ; Did Lock manager gave us old value

RMOSHARE
V04-000

SHARING ROUTINES

RM\$INIT_GBSB - Allocate and init the GBS

N 6

16-SEP-1984 00:37:46

5-SEP-1984 16:22:33

VAX/VMS Macro V04-00

[RMS.SRC]RMOSHARE.MAR;1

Page 20
(8)

E6	13	0334	904	BEQL	20\$
		0336	905	RMSERR	ENQ,R1
FCC2'	30	0338	906	BSBW	RMSMAPERR
E3	11	033E	907	BRB	25\$
		0340	908		

; Yes, treat as alternate success
; default to ENQ error for RMSMAPERR
; go map the error
; and return

0340 910 .SBTTL Global Buffer Section locking routines
 0340 911
 0340 912 :++
 0340 913 :
 0340 914 : RM\$RAISE_GBS_LOCK
 0340 915 : RM\$LOWER_GBS_LOCK
 0340 916 :
 0340 917 : These routines modify the lock mode on a global buffer section.
 0340 918 :
 0340 919 : Calling Sequence:
 0340 920 :
 0340 921 : BSBW RMSRAISE_GBS_LOCK - Get an exclusive lock on the section
 0340 922 : BSBW RM\$LOWER_GBS_LOCK - Get a concurrent read lock on the section
 0340 923 :
 0340 924 : Input Parameters:
 0340 925 :
 0340 926 : r9 ifab/irab address
 0340 927 :
 0340 928 : Implicit Inputs:
 0340 929 : none
 0340 930 :
 0340 931 : Output Parameters:
 0340 932 :
 0340 933 : r0 status code
 0340 934 :
 0340 935 : Implicit Outputs:
 0340 936 : none
 0340 937 :
 0340 938 : Completion Codes:
 0340 939 :
 0340 940 : suc, enq
 0340 941 :
 0340 942 : Side Effects:
 0340 943 :
 0340 944 : Could stall
 0340 945 :
 0340 946 :--
 0340 947 :
 50 05 D0 0340 948 RMSRAISE_GRS_LOCK:: : Convert GBS lock to EX.
 03 11 0340 949 MOVL #LCK\$K_EXMODE,R0 : Place lock mode in R0.
 0343 950 BRB TAKE_GBS_LOCK
 0345 951
 50 00 D0 0345 952 RMSLOWER_GBS_LOCK:: : Convert GBS lock to NL.
 0345 953 MOVL #LCK\$K_NLMODE,R0 : Place lock mode in R0.
 0348 954
 043E 8F BB 0348 955 TAKE_GBS_LOCK:
 55 50 D0 034C 956 PUSHR #^M<R1,R2,R3,R4,R5,R10>
 5A 59 D0 034F 957 MOVL R0,R5 : Save lock value
 0352 958 MOVL R9,R10 : Move ifab into r10 for stall
 0352 959 ASSUME <IFB\$C_BID&1> EQ 1 : in case it's really irab
 0352 960 ASSUME <IRB\$C_BID&1> EQ 0
 0352 961 ASSUME IFB\$B_BID EQ IRB\$B_BID
 03 08 AA E8 0352 962 BLBS IFB\$B_BID(R10),10\$-
 5A 6A D0 0356 963 MOVL IRB\$L_IFAB_LNK(R10),R10 : Do we have a ifab or irab
 54 7C AA D0 0359 964 10\$: MOVL IFB\$L_GBSB_PTR(R10),R4 : Get ifab
 43 13 035D 965 BEQL 50\$: Get gbsb
 035F 966 : If there is no global buffer secti : then ignore the request

			035F	967	:			
			035F	968	:	See if we already have the lock being requested, if so exit		
			035F	969	:			
			035F	970				
0A A4	55	91	035F	971	CMPB	R5, GBSB\$B_CURMODE(R4)		
3D	13		0363	972	BEQL	50\$		
			0365	973				
			0365	974	:			
			0365	975	:	Convert the lock on the global buffer section		
			0365	976	:			
			0365	977				
FC98'	30		0365	978	20\$: BSBW	RM\$SETEFN	: Allocate event flag.	
01	BA		0368	979	POPR	#^M<R0>	: Get the flag.	
			036A	980	SENQ_S	EFN = R0,-		
			036A	981		LKMODE = R5,-		
			036A	982		LKSBL = GB\$BSL_LKSBL(R4),-		
			036A	983		FLAGS = #LCKSM_SYNCSTS!LCKSM_SYSTEM!LCKSM_CONVERT!LCKSM_VALBLK,-		
			036A	984		ASTADR = RM\$STA[LAST],-		
			036A	985		ASTPRM = R9		
			038A	986				
0689	8F	1D 50	E9	038A	987	BLBC	R0,70\$: We ok?
		50	B1	038D	988	CMPW	R0,#SSS_SYNCH	: Did it complete?
		03	13	0392	989	BEQL	30\$: YES so do not stall
FC69'	30	0394		0390	990	BSBW	RM\$STALL_LOCK	: Stall for the lock
		0397		991	30\$:			
50	2C A4	3C	0397	992	MOVZWL	GBSB\$W_STATUS(R4),R0	: Get the final status	
0C	50	E9	039B	993	BLBC	R0,70\$: Branch on failure	
0A A4	55	90	039E	994	40\$:	MOVBL	R5, GBSB\$B_CURMODE(R4)	: Save the current lock value
043E	8F		03A2	995	50\$:	RMSSUC		: Signal success
			03A5	996	60\$:	POPR	#^M<R1,R2,R3,R4,R5,R10>	: Restore registers
		05	03A9	997		RSB		: Return
			03AA	998				
			03AA	999				
			03AA	1000	70\$:			
50	0E0A	8F	B1	03AA	1001	CMPW	#SSS_DEADLOCK,R0	: Was it deadlock?
	B4	13	03AF	1002	BEQL	20\$: Try it again if it was.	
50	09F0	8F	B1	03B1	1003	CMPW	#SSS_VALNOTVALID,R0	: Did Lock manager gave us old value
	E6	13	03B6	1004	BEQL	40\$: Yes, treat as alternate success	
			03B8	1005	RMSERR	ENQ,R1	: Default to ENQ error for RMSMAPERR	
FC40'	30	03BD	1006		BSBW	RMSMAPERR	: Go map the error	
E3	11	03C0	1007		BRB	60\$: and return	
		03C2	1008					

03C2 1010 .SBTTL RM\$RLS_GBSB - Deallocate the GBSB and dequeue the lock on it
 03C2 1011
 03C2 1012 :++
 03C2 1013 :
 03C2 1014 RMSRLS_GBSB
 03C2 1015 :
 03C2 1016 This routine deallocated the GBSB and then dequeues the lock it had on
 03C2 1017 the global buffer section. The dequeue also writes out the lock value
 03C2 1018 block to the lock manager.
 03C2 1019
 03C2 1020 Calling Sequence:
 03C2 1021 BSBW RM\$RLS_GBSB
 03C2 1022
 03C2 1023
 03C2 1024 Input Parameters:
 03C2 1025
 03C2 1026 r9 irab/ifab address
 03C2 1027
 03C2 1028 Implicit Inputs:
 03C2 1029 none
 03C2 1030
 03C2 1031 Output Parameters:
 03C2 1032
 03C2 1033 r0 status code
 03C2 1034
 03C2 1035 Implicit Outputs:
 03C2 1036 none
 03C2 1037
 03C2 1038 Completion Codes:
 03C2 1039
 03C2 1040 suc, deq
 03C2 1041
 03C2 1042 Side Effects:
 03C2 1043
 03C2 1044 none
 03C2 1045
 03C2 1046 :--
 03C2 1047
 0412 8F BB 03C2 1048 RMSRLS_GBSB::
 5A 59 D0 03C2 1049 PUSHR #^M<R1,R4,R10>
 03C6 1050 MOVL R9,R10 ; Move ifab into r10
 03C9 1051
 03C9 1052 ASSUME <IFB\$C_BID&1> EQ 1 ; in case it's really irab
 03C9 1053 ASSUME <IRB\$C_BID&1> EQ 0
 03C9 1054 ASSUME IFB\$B_BID EQ IRB\$B_BID
 03C9 1055
 03 08 AA E8 03C9 1056 BLBS IFB\$B_BID(R10),10\$; Do we have a ifab or irab
 5A 6A D0 03CD 1057 MOVL IRB\$L_IFAB_LNK(R10),R10 ; Get ifab
 54 7C AA D0 03D0 1058 10\$: MOVL IRB\$L_GBSB_PTR(R10),R4 ; Get gbsb
 19 13 03D4 1059 BEQL 30\$; Skip if none.
 0088 CA D4 03D6 1060 20\$: CLRL IFB\$L_GBH_PTR(R10) ; Indicate that global section is go
 03DA 1061 SDEQ_S LKID = GBSB\$L_LOCK_ID(R4),- ; Dequeue the lock,
 03DA 1062 VALBLK = GBSB\$L_LKSBL8(R4) ; writing out the lock value block.
 7C AA D4 03E9 1063 CLRL IFB\$L_GBSB_PTR(R10) ; Indicate that GBSB is gone.
 FC11 30 03EC 1064 BSBW RM\$RETBLK1 ; Deallocate the GBSB, address in R4
 0412 8F BA 03EF 1065 30\$: POPR #^M<R1,R4,R10>
 05 03F3 1066 RSB ; Return to caller.

RMOSHARE
V04-000

SHARING ROUTINES

RMSRLS_GBSB - Deallocate the GBSB and de

E 7

16-SEP-1984 00:37:46 VAX/VMS Macro V04-00

5-SEP-1984 16:22:33 [RMS.SRC]RMOSHARE.MAR;1

Page 24
(10)

03F4 1067

03F4	1069		.SBTTL VALIDATE_EBK_HBK	
03F4	1070	++		
03F4	1071		VALIDATE_EBK_HBK	
03F4	1072			This routine reads the record attributes for a file from the disk to see if the value block or the disk contains the most up to date information, and move the latest information into the SFSB for use in the value block and ifab.
03F4	1073			
03F4	1074			
03F4	1075			
03F4	1076			
03F4	1077			
03F4	1078			
03F4	1079			
03F4	1080			
03F4	1081			
03F4	1082			
03F4	1083			
03F4	1084			
03F4	1085			
03F4	1086			
03F4	1087			
03F4	1088			
03F4	1089			
03F4	1090			
03F4	1091			
03F4	1092			
03F4	1093			
03F4	1094	--		
03F4	1095			
03F4	1096	\$NEWPSECT	RM\$RMSMISC	
0000000C	0000	1097		
6A	0000	1098	ATR_LIST_LEN = 12	
	0000	1099	BUF_LEN:.BYTE ATR_LIST_LEN+FHCLEN+8+FIBSC_LENGTH	
	0001	1100		
	0001	1101		
	0001	1102	Note that the buffer allocated in this routine contains the following	
	0001	1103	structures, all of which are used as parameters to the file system.	
	0001	1104	1. attribute list (last longword in list is a zero-longword)	
	0001	1105	2. record attributes buffer	
	0001	1106	3. fib descriptor (8 bytes)	
	0001	1107	4. fib	
	0001	1108		
	0001	1109		
043E 8F	BB	0001	VALIDATE_EBK_HBK:	
		0001	PUSHR #^M<R1,R2,R3,R4,R5,R10>	
		0005		
		1112		
		0005	ASSUME SFSBSL_EBK EQ <SFSBSL_HBK + 4>	
		1113	ASSUME FATSL_EFBLK EQ <FAT\$L_AIBLK + 4>	
		0005	ASSUME <iFBSC_BID&1> EQ 1	
		1114	ASSUME <iRBSC_BID&1> EQ 0	
		0005	ASSUME IFBSB_BID EQ IRBSB_BID	
		1115		
		0005		
		1116		
		0005		
		1117		
		0005		
		1118		
		0005		
		1119	MOVL R9,R10	: Move ifab into r10 for stall
5A 59	D0	0005	BLBS IFBSB_BID(R10),1\$: Do we have a ifab or irab
03 08 AA	E8	0008	1120	
5A 6A	D0	000C	MOV L IRBSL_IFAB_LNK(R10),R10	Get ifab
52 EE AF	9A	000F	1121	r2 = length of buffer to alloc
00000000'EF	16	0013	1122 1\$: MOVZBL BUF_LEN,R2	Get the space (returned in R1)
08 50	E8	0019	1123 JSB RMSGETSPC1	Error?
		001C	1124 BLBS R0,10\$	Yes, map it and return.
		1125	RMSERR DMÉ,R1	

61	00040016	008E	31	0021	1126	BRW	60\$			
04	A1	0C A1	D0	0024	1127	10\$:	MOVL	#CATR\$C RECATTR@16>+FHCLEN,(R1)	: First longword of attrib list	
53	22 A1	9E	002B	1128	MOVAB	ATR LIST LEN(R1),4(R1)		: Second longword points to buffer.		
63	40 8F	DE	0030	1129	MOVAL	FHCLEN+ATR LIST LEN(R1),R3		: r3 = fib desc address		
04	A3	08 A3	DE	0034	1130	MOVZBL	#FIBSC_LENGTH,(R3)		: fill in length field of desc	
	4F AA	9A	0038	1131	MOVAL	8(R3),4(R3)		: fill in address of fib		
	36 A3	90	003D	1132	MOVVB	IFBSB_AGENT_MODE(R10),-		: move agent mode into fib		
	54	51	D0	0040	1133		8+FIB\$B_AGENT_MODE(R3)			
00000000	'EF	16	0042	1134	MOVL	R1,R4		: Save address of allocated space.		
01	BA	0045	1135	JSB	RM\$SETEFN			: Get event flag.		
		004B	1136	POPR	#^M<R0>			: Put it in R0.		
		004D	1137		SQIO_S	EFN = R0,-		: Go read the header attributes		
		004D	1138			CHAN = IFBSW_CHNL(R10),-		: from disk.		
		004D	1139			FUNC = #IOS_ACCESS,-		: gio function code		
		004D	1140			IOSB = IFBS\$[IOS(R9),-		: io status block		
		004D	1141			ASTADR = RMSSTALLLAST,-		: ast address		
		004D	1142			ASTPRM = R9,-		: ast parameter		
		004D	1143			P1 = (R3),-		: fib descriptor address		
		004D	1144			P5 = R1		: attribute list address		
	32 50	E9	0072	1145	BLBC	R0,50\$: Did qio succeed?		
00000000	'EF	16	0075	1146	JSB	RM\$STALL_LOCK		: Yes, stall for io to complete.		
29 50	E9	007B	1147	BLBC	R0,50\$: Return if unsuccessful.			
04 A4	04 A4	OC	007E	1148	ADDL2	#ATR_LIST_LEN,R4		: Get address of buffer.		
08 A4	08 A4	10	9C	0081	1149	ROTL	#16,FAT\$L_HIBLK(R4),FAT\$L_HIBLK(R4)	: Rotate HBK.		
		9C	0087	1150	ROTL	#16,FAT\$L_EFBLOCK(R4),FAT\$L_EFBLOCK(R4)	: Rotate EBK.			
40 A3	08 A4	D0	008D	1151	MOVL	IFBS\$SFSB_PTR(R10),R3		: Put SFSB address in R3.		
	07	14	0091	1152	CMPL	FAT\$L_EFBLK(R4),SFSB\$L_EBK(R3)		: Is disk EOF mark higher?		
3C A3	04 A4	D1	0096	1153	BGTR	30\$: Yes, use disk info.		
	05	15	0098	1154	CMPL	FAT\$L_HIBLK(R4),SFSB\$L_HBK(R3)		: Is disk HBK mark higher?		
3C A3	04 A4	15	009D	1155	BLEQ	40\$: No, use value block values.		
	54	OC	7D	009F	1156	30\$:	MOVQ	FAT\$L_HIBLK(R4),SFSB\$L_HBK(R3)	: Move disk ebk/hbk to sfsb.	
52	FF55 CF	C2	00A4	1157	40\$:	SUBL2	#ATR_LIST_LEN,R4	: Prepare to give back space.		
00000000	'EF	16	00A7	1158	50\$:	MOVZBL	BUFLEN,R2	: Length of space to deallocate		
043E 8F	BA	00AC	1159			JSB	RMS\$RETSPC1	: Address in R4		
	05	00B2	1160	60\$:		POPR	#^M<R1,R2,R3,R4,R5,R10>	: Restore registers.		
		00B6	1161			RSB		: Return.		
		00B7	1162							
		00B7	1163			.END				

SHARING ROUTINES

H 7

16-SEP-1984 00:37:46 VAX/VMS Macro V04-00
5-SEP-1984 16:22:33 [RMS.SRC]RMOSHARE.MAR;1Page 27
(11)

\$\$PSECT_EP	= 00000000		FIBSW_FID_RVN	= 00000008
\$\$ARGS	= 0000000B		FIBSW_FID_SEQ	= 00000006
\$\$RMSTEST	= 0000001A		FTLS_BADEBKHBK	= FFFFFFFD4
\$\$RMS_PBUGCHK	= 00000010		FTLS_NOPARENT	= FFFFFFFFO
\$\$RMS_TBUGCHK	= 00000008		FWASQ_SHRFIL_LCK	= 00000198
\$\$RMS_UMODE	= 00000004		FWAST_FIBBUF	= 000001F4
SST1	= 00000000		GBSB\$B_BID	= 00000008
ALTSUC	00000221 R 01		GBSB\$B_BLN	= 00000009
ATRSC_RECATTR	= 00000004		GBSB\$B_CURMODE	= 0000000A
ATR_LIST_LEN	= 0000000C		GBSB\$C_BID	= 00000009
BAD_EBKHBK	0000022D R 01		GBSB\$C_BLN	= 00000044
BAD_EBKHBK1	00000234 R 01		GBSB\$L_ADDRESS	= 00000004
BUF_LEN	00000000 R 03		GBSB\$L_LKSB	= 0000002C
CHECK_SHARE_OPTIONS	00000154 R 01		GBSB\$L_LOCK_ID	= 00000030
DO_ENQ	00000072 R 01		GBSB\$Q_FILENAME	= 00000000
ENQ	000001DD R 01		GBSB\$T_RESNAM	= 0000000C
ENQS_ACMODE	= 00000028		GBSB\$W_NAME_LEN	= 00000000
ENQS_ASTADR	= 0000001C		GBSB\$W_STATUS	= 0000002C
ENQS_ASTPRM	= 00000020		HBKEBK	0000021A R 01
ENQS_BLKAST	= 00000024		IFB\$B_AGENT_MODE	= 0000004F
ENQS_EFN	= 00000004		IFB\$B_BID	= 00000008
ENQS_FLAGS	= 00000010		IFB\$B_JNLFLG2	= 000000A2
ENQS_LKMODE	= 00000008		IFB\$B_RFMOORG	= 00000050
ENQS_LKSB	= 0000000C		IFB\$C_BID	= 00000008
ENQS_NARGS	= 0000000B		IFB\$C_FHAEND	= 00000066
ENQS_PARID	= 00000018		IFB\$L_EBK	= 00000074
ENQS_PROT	= 0000002C		IFB\$L_GBH_PTR	= 00000088
ENQS\$RESNAM	= 00000014		IFB\$L_GBSB_PTR	= 0000007C
ENQERR	0000023B R 01		IFB\$L_HBK	= 00000070
ENQRET	00000228 R 01		IFB\$L_IOS	= 0000000C
ENQSUC	00000225 R 01		IFB\$L_PAR_LOCK_ID	= 00000080
EXE\$GL_SYSID_LOCK	***** X 01		IFB\$L_SFSB_PTR	= 00000078
FAB\$B_FAC	= 00000016		IFB\$V_RUP	= 00000002
FAB\$B_SHR	= 00000017		IFB\$W_CHNL	= 00000020
FAB\$L_FOP	= 00000004		INIT_SFSB	00000022 R 01
FAB\$M_BIO	= 00000020		IOS_ACCESS	= 00000032
FAB\$M_BRO	= 00000040		IRBSB_BID	= 00000008
FAB\$M_DEL	= 00000004		IRBS\$C_BID	= 0000000A
FAB\$M_GET	= 00000002		IRBSL_IFAB_LNK	= 00000000
FAB\$M_PUT	= 00000001		LCK\$K_CRMODE	= 00000001
FAB\$M_UPD	= 00000008		LCK\$K_EXMODE	= 00000005
FAB\$V_DEL	= 00000002		LCK\$K_NLMODE	= 00000000
FAB\$V_GET	= 00000001		LCK\$K_PWMODE	= 00000004
FAB\$V_PUT	= 00000000		LCK\$M_CONVERT	= 00000002
FAB\$V_SHRDEL	= 00000002		LCK\$M_CVTSYS	= 00000040
FAB\$V_SHRGET	= 00000001		LCK\$M_SYNCSTS	= 00000008
FAB\$V_SHRPUT	= 00000000		LCK\$M_SYSTEM	= 00000010
FAB\$V_SHRUPD	= 00000003		LCK\$M_VALBLK	= 00000001
FAB\$V_UFO	= 00000011		PSL\$C_EXEC	= 00000001
FAB\$V_UPD	= 00000003		RMSBUG	***** X 01
FAB\$VUPI	= 00000006		RMSGETBLK	***** X 01
FATSL_EFBLK	= 00000008		RMSGETSPC	***** X 01
FATSL_HIBLK	= 00000004		RMSGETSPC1	***** X 03
FHCLEN	= 00000016		RMSINIT_GBSB	00000292 RG 01
FIB\$B_AGENT_MODE	= 0000002E		RMSINIT_SFSB	00000000 RG 01
FIB\$C_LENGTH	= 00000040		RMSINIT_SFSB_IRB	000000EA RG 01
FIB\$W_FID_NUM	= 00000004		RMSLOWER_GBS_LOCK	00000345 RG 01

RMSLOWER_LOCK	000001A8	RG	01
RMSLOWER_SYSLOCK	00000198	RG	01
RMSMAPERR	*****	X	01
RMSRAISE_GBS_LOCK	00000340	RG	01
RMSRAISE_LOCK	0000018C	RG	01
RMSRESTORE_LOCK	00000180	RG	01
RMSRETBLOCK1	*****	X	01
RMSRETSPEC1	*****	X	03
RMSRLS_GBSB	000003C2	RG	01
RMSRLS_SFSB	0000025F	RG	01
RMSSAVE_FL	*****	X	01
RMSSETEFN	*****	X	01
RMSSTALLLAST	*****	X	01
RMSSTALL_LOCK	*****	X	01
RMSS_DME	= 000184D4		
RMSS_ENQ	= 0001C134		
RMSS_INCOMPNSHR	= 0001826A		
RMSS_SHR	= 000186B4		
RMSS_UPI	= 000187AC		
SET_LOCK	000001B2	R	01
SFSBSB_BID	= 00000008		
SFSBSB_CURMODE	= 0000000A		
SFSBSB_FAC	= 00000034		
SFSBSB_PREMODE	= 0000000B		
SFSBSB_SHR	= 00000035		
SFSBSL_BID	= 00000010		
SFSBSL_BLN	= 00000044		
SFSBSL_FIX_LEN	= 0000000A		
SFSBSL_ADDRESS	= 00000004		
SFSBSL_EBK	= 00000040		
SFSBSL_FAC_CODE	= 0000000C		
SFSBSL_HBK	= 0000003C		
SFSBSL_LKSB	= 0000002C		
SFSBSL_LOCK_ID	= 00000030		
SFSBSL_LVB	= 00000034		
SFSBSQ_FILENAME	= 00000000		
SFSBST_DEV_NAM	= 00000016		
SFSBST_RESNAM	= 0000000C		
SFSBSW_FID_NUM	= 00000010		
SFSBSW_FID_RVN	= 00000014		
SFSBSW_FID_SEQ	= 00000012		
SFSBSW_NAME_LEN	= 00000000		
SFSBSW_STATUS	= 0000002C		
SHRBITS	= 0000000F		
SHRERR	0000017A	R	01
SHREXT	00000176	R	01
SSS_DEADLOCK	= 0000E0A		
SSS_SYNCH	= 0000689		
SSS_VALNOTVALID	= 00009F0		
SYSSDEQ	*****	GX	01
SYSENQ	*****	GX	01
SYSSQIO	*****	GX	03
TAKE_GBS_LOCK	00000348	R	01
VALIDATE_EBK_HBK	00000001	R	03

+-----+
! Psect synopsis !
+-----+

PSECT name

	Allocation	PSECT No.	Attributes																	
ABS	00000000 (0.)	00 (0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE							
RMSRMSO	000003F4 (1012.)	01 (1.)	PIC	USR	CON	REL	GBL	NOSHR	EXE	RD	NOWRT	NOVEC	BYTE							
SABSS	00000000 (0.)	02 (2.)	NOPIC	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE							
RMSRMSMISC	000000B7 (183.)	03 (3.)	PIC	USR	CON	REL	GBL	NOSHR	EXE	RD	NOWRT	NOVEC	BYTE							

+-----+
! Performance indicators !
+-----+

Phase

Phase	Page faults	CPU Time	Elapsed Time
Initialization	30	00:00:00.09	00:00:00.84
Command processing	120	00:00:00.63	00:00:03.78
Pass 1	557	00:00:24.37	00:01:07.76
Symbol table sort	0	00:00:03.55	00:00:06.54
Pass 2	199	00:00:04.95	00:00:13.75
Symbol table output	20	00:00:00.19	00:00:00.56
Psect synopsis output	3	00:00:00.03	00:00:00.13
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	931	00:00:33.82	00:01:33.36

The working set limit was 1950 pages.

134597 bytes (263 pages) of virtual memory were used to buffer the intermediate code.

There were 130 pages of symbol table space allocated to hold 2407 non-local and 45 local symbols.

1163 source lines were read in Pass 1, producing 19 object records in Pass 2.

40 pages of virtual memory were used to define 39 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name

Macros defined

Macro library name	Macros defined
\$255\$DUA28:[RMS.OBJ]RMS.MLB;1	15
-\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	1
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	19
TOTALS (all libraries)	35

2620 GETS were required to define 35 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LI\$:RMOSHARE/OBJ=OBJ\$:RMOSHARE MSRC\$:RMOSHARE/UPDATE=(ENH\$:RMOSHARE)+EXECMLS/LIB+LIB\$:RMS/LIB

0320 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

RM0XPEN
LIS

RM0SETDID
LIS

RM0SHARE
LIS

RM0WILD
LIS

RM0XAB
LIS

RM0STALL
LIS